Wavelet-based system for recognition and labeling of polyhedral junctions

Gabriel Fernández
GE Medical Systems
Magnetic Resonance
P.O. Box 414, W-832
Milwaukee, Wisconsin 53201

Terrance Huntsberger, MEMBER SPIE
University of South Carolina
Department of Computer Science
Intelligent Systems Laboratory
Columbia, South Carolina 29208
E-mail: terry@cs.sc.edu

Abstract. Recognition of junctions in scenes containing polyhedral objects has been shown to be an effective means of encoding for model-based matching. However, noise and inconsistent lighting in the images limit the amount of useful information than can be extracted for junction identification. A wavelet-based system for polyhedral junction recognition that combines the wavelet-derived edges and line elements at multiple levels of resolution to produce a labeled image is introduced. The results of some experimental studies are also reported. [S0091-3286(98)00901-5]

Subject terms: recognition techniques; wavelets; lifting scheme; machine vision; reconstruction; key points; edges; contours; interpretation.

Paper ART-122 received June 2, 1997; revised manuscript received Aug. 12, 1997; accepted for publication Aug. 22, 1997.

1 Introduction

People might ask how it is that they can see objects in depth, meaning that human vision is ''three dimensional.'' The most common reply is that humans use the difference between the images in our left and right eyes to judge depth. Many techniques have been proposed to simulate human vision with machines and to find those points that human eyes seem to use as references to build more complex models. If a machine is able to see well, then we tend to call it intelligent. The area of artificial intelligence (AI) uses computers and their computational ideas and methods to study intelligence. AI offers a new perspective and a new methodology to make computers ''intelligent.'' Machine vision, however, must address many problems:

1. What information should be extracted?
2. How should this information be extracted?
3. How should this information be represented?
4. How should this information be used?

This paper intends to present solutions to some of the problems mentioned above. We address the problem of 2-D intermediate-level visual processing and derive a robust and efficient method that processes a raw image signal over several layers of abstraction to produce meaningful intermediate level structures. In addition, both the primitive features and the aggregated structures are integrated into a symbolic representation.

2 Multiresolution Edge Detector

Our goal is to analyze light intensity data from given input images to describe models in the original scene. To simplify the process, we only consider trihedral objects, i.e., solid polyhedra bounded by planar faces where exactly three faces meet at each vertex.

2.1 *Features*

Features, such as edges, lines, points, and regions, must be previously extracted to generate object hypotheses from a symbolic image description. The feature extraction procedure usually uses a local operator and does not use any specific shape information or contextual scene knowledge. This process is often referred to as (low-level) image segmentation. The problem of image segmentation is still unsolved, mainly because it is not independent of the overall task and there is no guarantee that the object parts (e.g., boundaries) are completely visible.

2.2 *Edge Maps*

A better approach is to derive a partial symbolic representation as opposed to a complete image segmentation. This process would extract points, edges, lines, and regions from gray-level and color images. Some operators are based on a simple and isolated model feature such as an ideal step edge[1] or a constant intensity region.[2] However, the assumption that image features appear in isolation and that they belong to a single class is often invalid. We can see that edges, lines, and points often interact with each other through composite edges or at image junctions.[3]

2.3 *Edge Detection Methods*

Intensity discontinuities are considered one of the primary image features that enable a scene to be segmented into meaningful parts. Many methods have been defined to suitably detect edge maps and some of them (the ones we used as inspiration for our multiresolution method) are briefly described here.

2.3.1 *Masks*

Point and line detection can be implemented by using simple masks that detect discontinuities, which is where these features are assumed to be located.[4,5]

2.3.2 *First and second derivatives*

We would like to take advantage of the fact that brightness changes more rapidly in the edges than in other places. Such behavior can be described in a better way with the first and second derivatives. The first derivative of brightness has a maximum in the edge, and the second derivative has a zero in that maximum. The generalization of the first derivative to 2-D is given by the gradient ($\nabla \mathbf{f}$).

2.3.3 *Gaussian filters*

Some other methods find edges in smoothed and subsampled images rather than the original images to avoid noise features and unreliable edges. The first edge-detection method using a Gaussian-shaped low-pass filter is described as follows:[6]

1. Filter signal using a Gaussian-shaped function:

   $$h(x,y)=\exp[-(x^2+y^2)/2\pi\sigma^2],$$
   $$H(u,v)=2\pi^2\sigma^2\ \exp[-\pi\sigma^2(u^2+v^2)/2],$$

   where σ determines the cutoff frequency, with a larger σ corresponding to a lower cutoff frequency.
2. Use an edge detection method.

2.3.4 *Combining ideas*

The previously described methods, as well as many other edge detection methods, work well for the specific cases and problems they were created to solve. We usually base our method on a specific edge model. A more complete approach would deal with several models under different situations to give better and more complete edge maps. In the next sections, we show how wavelets can be used to acquire a multiresolution representation of the input signal, how to automatically apply a smoothing procedure by defining a smoothing function (such as a Gaussian), and how to calculate the first and second derivatives to calculate the gradients and represent edge pixels through several levels of resolution.

2.4 *Wavelets*

It has become impossible to give the definition of a wavelet.[7] Basically, definitions become obsolete from one day to the next due to fast growth of the research field and the high rate at which contributions are made. A very vague definition, but one that at least includes three of the main features of wavelets is: ''wavelets are building blocks that can quickly decorrelate data.''

Let us analyze these three properties. First, wavelets are building blocks for general data sets or functions. In mathematical terms, we say that they form a basis or, more generally, a frame. By this we mean that each element of a general class can be written in a stable way as a linear combination of the wavelets. If the wavelets are denoted by ψ_i and the coefficients by γ_i, a general function f can be written as

$$f=\sum_i \gamma_i\psi_i .$$

Second, wavelets have the power to decorrelate. This means that the representation of the data in terms of the wavelet coefficients γ_i is somehow more ''compact'' than the original representation. In information-theory terms, we say that the entropy in the wavelet representation is smaller than the original representation, and in approximation-theory terms, we say that we want to get an accurate approximation of f by only using a small fraction of the wavelet coefficients.

This decorrelation is obtained by constructing wavelets that already resemble the data we want to represent. In other words, we would like the wavelets to have the same correlation structure as the data. For instance, most signals we encounter every day have correlation in both space and frequency, i.e., samples that are spatially close are much more correlated that ones that are far apart, and frequencies often occur in bands. If we want to analyze and represent such signals, we need wavelets that are local in space and frequency. Typically, this is achieved by building wavelets that have compact support (localization in space), that are smooth (decay toward high frequencies), and that have vanishing moments (decay toward low frequencies).

Third and last, we want to quickly find the wavelet representation of the data, i.e., we want to switch between the original representation of the data and its wavelet representation in a time proportional to the size of the data [something that in algorithmic-complexity terms is known as $O(n)$]. The fast decorrelation power of wavelets is the key to applications such as data compression, fast data transmission, noise cancellation, signal recovery, and fast numerical algorithms.

2.5 *Lifting Scheme*

The basic idea behind lifting is that it provides a simple relationship between all resolution analyses that share the same low-pass filter or high-pass filter.[8] The low-pass filter provides the coefficients of the refinement relation, which entirely determines the scaling function. The high-pass filter provides the coefficients that enable the linear combination of several scaling functions to find the wavelet. Lifting can be used to effortlessly custom-design wavelets. One, for example, could derive a family of biorthogonal wavelets associated to the interpolating Deslauriers-Dubuc scaling functions[9] using lifting.[10]

The ''lifting scheme'' [7,8,10–15] is a new approach for the construction of families of wavelets that are independent of the Fourier transform. Constructing wavelets using lifting consists of three simple phases or stages: the first one SPLITS the data into two subsets, even and odd, the second one calculates the wavelet coefficients (high pass) as the failure to PREDICT the odd set based on the even, and finally the third one UPDATES the even set using the wavelet coefficients to compute the scaling function coefficients (low pass). The PREDICT phase ensures polynomial cancellation in the high pass (vanishing moments of the dual wavelet) and the UPDATE phase ensures preservation of moments in the low pass (vanishing moments of the primal wavelet).

The advantages of lifting are numerous:

1. Lifting allows for an in-place implementation of the fast wavelet transform, a feature similar to the fast Fourier transform.[7]
2. It is particularly easy to build wavelet transforms that map integers to integers using lifting.[15] These transforms are particularly useful for hardware implementation and for lossless image coding.
3. Lifting enables the construction of wavelets entirely in the spatial domain, i.e., without making use of the Fourier transform. This means that it can be used to build wavelets that are not necessarily translates or dilates of one function. These wavelets are known as ''second-generation wavelets'' and typical examples are wavelets adjusted to weight functions, irregular samples,[14] the sphere,[13] or manifolds. This also enables an easy way to introduce wavelets, which is particularly useful for people without a strong mathematical background.[14]
4. Every transform built with lifting is immediately invertible where, the inverse transform has exactly the same computational complexity as the forward transform.[11,16]
5. Lifting allows for adaptive wavelet transforms, i.e., one can start the analysis of a function from the coarsest levels and then build the finer levels by refining only in the areas of interest.[13]
6. Lifting exposes the parallelism inherent in a wavelet transform. All operations within one lifting step can be done entirely in parallel. The order of the lifting operations in the only part required to be sequential.

For our particular problem, feature extraction, we used wavelets generated by LiftPack,[11] which is a C library of biorthogonal wavelets with n vanishing moments. Refer to Ref. 11 for more details about LiftPack and the implementation of the lifting scheme.

2.6 *Multiresolution Edge Representation*

Our model of a multiresolution edge representation is based on the local maxima of the wavelet transform. The wavelet transform of a signal is a multiresolution decomposition that is well localized in space and frequency.[17] The multiresolution edge representation of signals was first described by Mallat[18] and has evolved in two forms, based on multiresolution zero-crossings and multiresolution gradient maxima, respectively. The latter, which is used in this paper, was developed by Mallat and Zhong.[19] We give only a brief review of the multiresolution representation for 2-D images. For a detailed description, see Refs. 19 and 20.

Consider two oriented wavelets that are constructed as the partial derivatives of a smoothing function $\varphi(x,y)$:

$$\psi^1(x,y)=\frac{\partial}{\partial x}\varphi(x,y) \quad \text{and} \quad \psi^2(x,y)=\frac{\partial}{\partial y}\varphi(x,y). \tag{1}$$

Assume an image is a differentiable 2-D function $f(x,y) \in \mathbf{L}^2(\mathbf{R}^2)$. The associated 2-D dyadic wavelet transform (WT) of an image f at scale 2^j, at position (x,y) and in orientation k is defined as

$$\mathscr{W}^k_{2^j} f(x,y)=f*\psi^k_{2^j}(x,y), \quad k=1,2, \tag{2}$$

with $\psi^k_{2^j}(x,y)=2^{-2j}\psi^k(2^{-j}x,2^{-j}y)$.

The WT defined by Eq. (2) produces a sequence of vector fields indexed by level of resolution. They are the gradient of $f(x,y)$ smoothed by $\varphi(x,y)$ at dyadic scales, or the multiresolution gradients:

$$\begin{aligned}\nabla_{2^j} f(x,y) &\equiv [\mathscr{W}^1_{2^j} f(x,y),\ \mathscr{W}^2_{2^j} f(x,y)] \\ &= \frac{1}{2^{2j}}\nabla(\varphi_{2^j}*f)(x,y) \\ &= \frac{1}{2^{2j}}\nabla f*\varphi_{2^j}(x,y).\end{aligned} \tag{3}$$

The multiresolution gradient representation of f is complete, because the WT defined by Eq. (2) is invertible. For a multiresolution gradient representation over a finite number of scales, $0 \leqslant j \leqslant J$, it is also necessary to include $S_{2^j} f(x,y)$, which is the smoothed version of f at the coarsest scale, 2^J.

The multiresolution edge representation is built on the multiresolution gradient representation. For this purpose, it is convenient to represent the multiresolution gradient in magnitude-angle pairs, $\rho_{2^j} f(x,y), \theta_{2^j} f(x,y)$, where the magnitude $\rho_{2^j} f(x,y)$ and angle $\theta_{2^j} f(x,y)$ are defined by

$$\rho_{2^j} f(x,y)=\{[\mathscr{W}^1_{2^j} f(x,y)]^2+[\mathscr{W}^2_{2^j} f(x,y)]^2\}^{1/2} \tag{4}$$

and

$$\theta_{2^j} f(x,y)=\arctan\left[\frac{\mathscr{W}^2_{2^j} f(x,y)}{\mathscr{W}^1_{2^j} f(x,y)}\right]. \tag{5}$$

A point (x,y) is considered a multiresolution edge point at scale 2^j if the magnitude of the gradient, $\rho_{2^j} f$, attains a local maximum there along the gradient direction $\theta_{2^j} f$. For each scale 2^j, we collect the edge points along with the corresponding values of the gradient (i.e., the wavelet transform values) at that scale. The resulting local gradient maxima set at scale 2^j is then given by

$$\begin{aligned}\mathscr{A}(f) = \Big\{ &[(x_i,y_i);\nabla_{2^j} f(x_i,y_i)] \\ &\times \Big| \begin{matrix}\rho_{2^j} f(x_i,y_i) \text{ has local maximum at} \\ (x_i,y_i) \text{ along the direction } \theta_{2^j} f(x_i,y_i)\end{matrix} \Big\}.\end{aligned} \tag{6}$$

For a J-level 2-D WT, the collection

$$\{S_{2^j} f(x,y),[\mathscr{A}_{2^j}(f)]_{1 \leqslant j \leqslant J}\} \tag{7}$$

is called a multiresolution edge representation of the image $f(x,y)$.

This method enables us to extract an object's features through different resolutions. However, we want to get subsampled versions of the original image for further calculations at different resolutions. The multiresolution edge representation proposed by Mallat does not subsample through

different scales. We use instead the fast lifted wavelet transform (FLWT) to adapt the multiresolution edge representation to our needs.

2.7 *Noise Estimation*

The multiresolution edge maps obtained with the local maxima operator described consist of both noise and true image features. Therefore, a thresholding operation must be applied to the edge maps to separate the noise features from the true image features. This threshold is often set manually by a trial-and-error procedure to produce a ''visually clean'' edge representation. It is, however, necessary to know the noise characteristics to determine threshold levels.[21]

It has been demonstrated[22,23] that the edge map threshold can be derived from the distribution of gradient magnitudes in the image. A method derives an automatic edge detection threshold by estimating the noise in the image. Let us establish that the original distribution of the gradient magnitudes consists of two components: a noise and a signal component. Since noise in images is usually additive, white Gaussian noise appears as the most prevalent model to use. If we assume white Gaussian noise, the gradient magnitudes of the noise are Rayleigh distributed. Due to the fact that the gradient magnitudes of the signal component are often considerably stronger than the ones from the noise component, the signal without the noise affects mainly the tail of the distribution. We want, therefore, to estimate the peak of the Rayleigh distribution from the original distribution. It can be estimated by computing the mode of the original distribution (Ref. 24, Chap. 13.3). Given an estimate ξ of the peak of the Rayleigh distribution, the edge detection threshold τ can be chosen to exclude a certain amount of edge pixels due to noise.[23] To remove noise edges with a risk probability of s, an edge detection threshold

$$\tau = \xi(-2 \ln s)^{1/2} \tag{8}$$

should be used. We use a probability risk of $s = 0.1\%$ or equivalently a threshold $\tau \approx 3.7169\xi$ for all experiments (unless we say otherwise). This threshold will enable us to remove noise features. If weak and unreliable edges should also be removed, a higher threshold value must be selected. At this time, however, we do not have enough information to decide which features are or are not relevant. This process is discussed later

3 Multiresolution Edgel Aggregation

Intensity discontinuities are considered one of the primary image features that enable a scene to be segmented into meaningful parts. Many methods have been defined to suitably detect edge maps. However, these edge maps have deficiencies: fragmentation, gaps at junctions, and clutter and faulty connections. Also, object boundaries are not guaranteed to be contrast defined. Additional processing is needed to obtain more complete and unambiguous boundary definitions that account for more global relationships among image features.[25]

3.1 *Contour Graph*

Edge detectors based on a local operator extract image features in a local neighborhood without using specific shape information or scene knowledge. They are unorganized and represent isolated image events. The feature extraction procedure must also organize these local features to a more complete symbolic image description. In our case, the edge detector produces an edge map in which each pixel consists of its local strength and orientation. Thus, the low-level organization consists of linking edge pixels to create a more complete contour representation—a contour graph. The more specific task of grouping these extracted contours to complete objects is not integrated in the work of this paper, but it is mentioned that coplanar grouping and segment stereo matching can be used to build full 3-D representations of the objects. Once we have found points that belong to an edge, we must link them together to find the actual edge. We use the method described in Ref. 3 for its unique way of dealing with different object features in the same algorithm.

3.2 *Key Points*

We have shown how to extract 1-D features such as edge/line points. However, even though those points might form meaningful boundaries, we do not really know how the points are related to each other and to which boundaries they belong. We could use a linking procedure with the gradient magnitude and orientation to construct contours or edges, but these 2-D features alone are not enough to give a complete symbolic description of an object. In addition to edges, other 2-D image features, such as junctions, corners, and line-ends represent another class of important information that can serve as the definition of object boundaries. First, they help to divide boundaries in a better way and help to extract meaningful shape decompositions. Also, corner and junction features together with the directions of their constituent components often characterize objects in a better way than edge fragments or corner and junction features alone. Second, many 2-D features occur in situations of occlusion and they can serve to indicate object contours even if the contrast is vanishing or null. We refer to these 2-D features as key-points and they must be included in the symbolic representation of objects.

The extraction of 2-D features is more straightforward than the corresponding nonmaximum suppression procedure for 1-D features. However, no method provides an entire solution to the classification of 2-D features, for example in L-, I-, T-, Y-, and X-junctions. In general-case methods, key-points are usually defined as strong 2-D intensity variations, i.e., the signal not only varies in one direction, but also in other directions. One could extract key-points by analyzing the first and second derivatives in the direction of modulus channels. Model-based methods, on the contrary, work with a specific corner model looking for areas that best match the model. If the classification is trivial with the model-based feature detectors, then the localization and detection tasks are by no means trivial. A combined approach could alleviate these problems, in that the key-points are used to initialize a model-based approach.

3.3 *Edgel Aggregation Algorithm*

An independent representation of key-points can help in aggregating local edge evidence to larger, coherent pieces of contour. The method represents the contrast-defined image features as a collection of meaningful parts, subdivided at locations of high key-point evidence. Even though this idea is not new, previous implementations used postprocessing of binary edge maps to achieve segmentation, rather than using independent representations of 2-D features. Key-points have an active role making contours converge onto them and reducing problems, especially at bifurcations. The natural relationship between edges and key-points can provide a more accurate definition of the image features.

We describe a general purpose algorithm that is able to represent, at different levels of resolution, edges and/or lines completely and accurately as well as their connections at corners, junctions and other important 2-D features. Considering the fact that edge and line operators usually produce unconnected pixel maps, the multiresolution edgel aggregation algorithm should also be able to bridge these small gaps (1 to 2 pixels). An additional requirement, is that the the algorithm should be independent of the edge/line operator.

The required data for the multiresolution edgel aggregation algorithm consists of those feature maps that multiresolution edge or line operators produce, i.e., the multiresolution magnitude, edge, and local orientation maps. Additional maps, such as the multiresolution type classification, general edge quality, and key-point maps, are optional and will be incorporated only if they are supplied.

Three major tasks must be addressed in the design of a general purpose edgel aggregation algorithm: where to start, how to link, and where to stop. These tasks have to be repeated at every level of resolution in order to obtain the multiresolution representation. The solutions to these issues are presented next.

3.3.1 *Where to start*

Linking is a sequential process. Therefore, it is important to aggregate the significant contours before the weaker ones. All edge/line pixels are assigned a start-point value that reflects their suitability to serve as seeds for the aggregation process. The ''optimal'' start-point is defined as the point which has two neighboring edge pixels of the same type (edge/line), is located in a region with homogeneous local orientation, and has a pure 1-D general edge profile. This means that pixels close to significant 2-D image features should not be considered because the local orientation is not reliable. See Refs. 26 and 3 for a detailed description of the process. The algorithm then picks the current best start-point from a sorted list of start-points. All the start-points on and along an established contour are invalidated or eliminated to prevent multiple aggregation of the same contour. The start-points can further be separated into edge and line start-points to allow the algorithm to deal with either only lines, only edges, or both together.

3.3.2 *How to link*

If the algorithm is to bridge small gaps in the edge map, a traditional 3×3 pixel neighborhood is not sufficient. One must extend the search region to solve this problem. However, the search region cannot extend too far because the computations involved increase dramatically. The next pixel is found by evaluating the pixels contained in a directed search region, which extends 2 pixels from the current position. Each directed search region consists of nine contiguous paths. A primary selection among these paths is based on simple criteria. The remaining paths are then ranked according to magnitude strength and orientation similarity and the best one is retained. Whenever the classification map is available (edge or line), the selection of the best path depends on the type consistency between the path and that of the start-point.[3] Finally, the selected path determines a new position and a new updated linking direction.

3.3.3 *Where to stop*

Key-points are used as a suitable stop condition for the aggregation process.[25] This enables us to divide contours into meaningful subsegments. The use of key-points also solves/reduces the problem of finding the correct link at junctions because the contours converge on them. The problem of imperfect localization of image features is treated by using a catch region of 3×3 pixels around each key-point. If the edgel aggregation algorithm converges onto a catch region it may connect the current contour to the key-point. Given the fact that key-point maps are not always available, additional termination conditions are used for robust performance. These conditions are when a contour segment occludes with an already established contour, occludes with itself, or when there are no more pixels to link.

Figure 1 shows all the component of the multiresolution edgel algorithm and how they interact.

3.4 *Postprocessing*

Considering that we are analyzing polyhedral objects and want a compact symbolic representation, we would like to obtain the maximum number of long and straight contours, while at the same time keeping the number of curved contours and endpoints as low as possible. This will be the main task of the postprocessing stage. The secondary task is to reduce the number of spurious contour segments for a much cleaner representation.

The 1-D and 2-D feature extraction procedures are domain-independent because they do not use any shape or contextual information. This is good for general applications. However, we are dealing with man-made or regular objects. We now introduce shape information—the straight contour—for a more customized processing.

The reorganization procedure depends on two main components: curve partitioning and the definition of straight, curved, and closed contours.

3.4.1 *Curve partitioning*

We propose to use an iterative split and merge algorithm (Ref. 27, Chap. 4) because of its simple implementation, speed, and reliable performance even on smoothly changing curves. Given a contour, the algorithm consists of four steps:

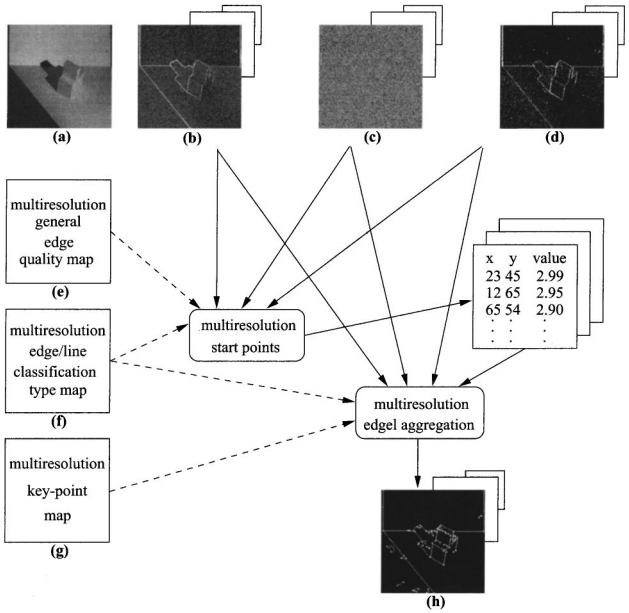


Fig. 1 Multiresolution edgel aggregation showing the required (solid line) and optional (dashed line) input for every module of the multiresolution edgel algorithm: (a) the input gray-scale image, (b) the multiresolution operator magnitude map, (c) the multiresolution local orientation map, (d) the multiresolution edge map, (e) the multiresolution general edge quality map, (f) the multiresolution type classification map, (g) the multiresolution key-point map, and (h) the resulting multiresolution symbolic representation.

1. Construct the line between the contour's endpoints and measure the maximum deviation D of the curve from that line.
2. If the absolute deviation exceeds some threshold T, split the curve at that point, and replace the original line with two new straight-line approximations between the old points and the new one.
3. Repeat recursively steps 1 and 2 for each new contour segment until all the segments have small enough deviations.
4. Finally, test consecutive segments to see if they can be merged into a single straight line, without exceeding the threshold T on deviation.

The threshold T need not to be a constant (typically 2 to 3 pixels), it can also be a function $L(d)$ of the Euclidean distance d between the two endpoints of the current subsegment. A variable threshold allows the algorithm to consider contours with a given maximum deviation as a straight line if the distance between the endpoints is large enough. In our case, L is defined as follows[3]:

$$L(d)=\begin{cases}1+\gamma\,\log_{10} d: & d\geqslant 1\\ 1: & \text{otherwise,}\end{cases} \tag{9}$$

where the constant 1 sets the lower bound for the tolerance and the parameter γ controls its increase, and γ is set to 1 by default. The fourth component of the algorithm is the merging procedure. The algorithm starts with the two segments that have the largest sum of distances between the endpoints. This guarantees that the algorithm always constructs straight lines as long as possible.

3.4.2 *Curve labeling*

For a robust symbolic representation, it is necessary to correctly classify the contours. We classify contours as closed, curved, and straight.[3] A contour is closed if it has the same endpoint at both ends. Open contours are classified as either straight or curved. A contour is straight if its length d is larger than some minimum d_l and if the maximum deviation, from the line between its end-points, is smaller than $L(d)$. The contour is labeled curved if at least one of the two conditions is not met. Notice that even though a contour has a maximum deviation of 0 pixels, it is labeled curved if its length is smaller than d_l.

We mentioned that the edge detection threshold should be selected according to an estimated level of image noise and not to a hard threshold. The reason for this was to avoid removing ''weak'' edges that could be part of stronger structures. Now, we want to reorganize all contours and endpoints to obtain contours as long and straight as possible, and how to remove obviously weak or spurious contour segments.

Many ideas have been proposed to reorganize contour segments and to remove weak edges. Most of them remove clearly significant structures and produce unnecessarily incomplete contour representations. A postprocessing procedure must clearly satisfy specific requirements (some times it is not even needed if the contour graph is ''clean'' enough).

The split- and-merge procedure consists of the following seven steps:

1. *Merge:* Suppress all endpoints with two connected contours and replace the two segments by a single one. If the contour is closed, the endpoint is not removed.
2. *Split:* A new set of break-points is computed for each contour.
3. *Prune:* Remove weak and curved contour segments if they are not connected to other contours at both ends. If a curved contour has an open end and its integrated gradient magnitude is lower than the median gradient magnitude among all contours at its level of resolution, the segment is pruned. Repeat this procedure until there are no more weak and curved contour segments with and open end.
4. *Merge:* Same as step 1.
5. *Split:* Same as step 2.
6. *Merge:* Group consecutive curved contours to build longer curved contours.
7. *MultiMerge:* If a contour segment is missing from one or more levels of resolution, but it appears in two or more levels of resolution in the multiresolution symbolic representation, create a new contour segment in the levels in which the contour is missing.

Steps 1 and 2 group smaller straight and curved segments to longer straight contours. The output is a reorganized set

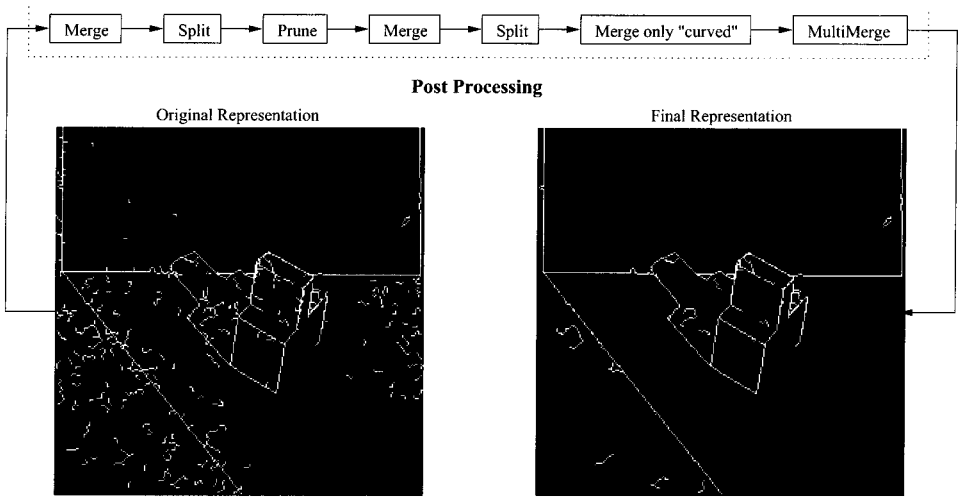


Fig. 2 Components and flow of the postprocessing algorithm: (a) contour graph resulting from the edgel aggregation algorithm and (b) final contour representation.

of contour graphs where each contour at every level of resolution is labeled either straight, closed, or curved. No single pixel is removed in these two steps. Step 3 removes all weak and curved contour segments that have unreliable links. Steps 4 and 5 obtain the longest possible straight contour segments. Step 6 reduces the number of contours and endpoints. Step 7 takes advantage of the redundant information of the multiresolution representation and complete unconnected contour graphs at certain levels of resolution.

The reason for applying the merge and split procedures before pruning the contours is that if the graph contains several weak and collinear ''curved'' segments they will all be removed. If these smaller segments are first grouped to longer straight contours they will always remain.[3] In addition, the multimerge step is effective only if a missing edge is exactly matched in the other levels.

Figure 2 shows the components and flow of the postprocessing algorithm described.

4 Conclusions and Future Work

A multiresolution feature extraction methodology has been proposed to efficiently find a symbolic representation of objects using wavelets and multiresolution analysis. The proposed algorithm is a combination of several powerful techniques and uses a large amount of information. The use of a large amount of information helps us to achieve redundancy, which helps us to obtain high-quality results. The multiresolution feature extraction algorithm consists of two main parts: the multiresolution edge detector and the multiresolution edgel aggregation algorithm. The former uses wavelets created using the lifting scheme, reorganizes the wavelet basis using the pyramid representation, and extracts edges using a wavelet local maxima operator. The latter uses an extension of an edgel aggregation algorithm to link contour segments at every level of resolution and to relate segments along different levels of resolution. The algorithm works for images of any size and for synthetic and real images. Results show that the proposed algorithm is robust and reliable under different conditions using both synthetic and real images. The sensitivity of the wavelet-based edge detector extracts edges in almost any type of lighting conditions. Small gaps are bridged thanks to an extended directional search mask. The automated noise detector and postprocessing procedure help to remove noise and spurious edges as well as to recuperate missing edges using the redundancy of the multiresolution representation.

Even though good results are obtained, the method is far from being finished. The algorithm does depend on the quality of the edge maps. The extended directional search mask and the postprocessing step are unable to handle highly fragmented and highly clustered edge maps. Results are unpredictable under these conditions. Adaptive procedures for noise removal and multiresolution edge completion seem to be the best options to solve these problems. Our work may also be complemented with segment stereo matching and coplanar grouping to generate a complete 3-D symbolic representation of objects. Even though we worked with only simple polyhedral objects, the proposed algorithm is perfectly adaptable to work with any type of objects.

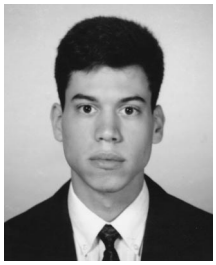
Acknowledgments

This work was partially supported under Army Research Office (ARO) grant no. DAAH049610326.

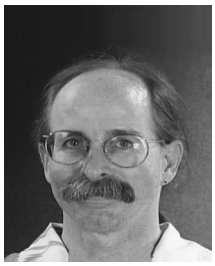
References

1. J. F. Canny, ''A computational approach to edge detection,'' *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(6), 679–698 (1986).
2. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Springer-Verlag (1982).
3. O. Henricsson, ''Analysis of image structures using color attributes and similarity relations, PhD Thesis, Swiss Federal Institute of Technology (ETH) Zürich, Diss. ETH. No. 11663 (1996).
4. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley (1993).
5. S. M. Smith and J. M. Brady, ''Susan—a new approach to low level image processing,'' Technical Report TR95SMS1c, Defense Research Agency, Farnborough, Hampshire, United Kingdom (1995).
6. D. Marr and E. Hildreth, ''Theory of edge detection,'' *Proc. R. Soc. Lond.* **B207**, 187–217 (1980).
7. W. Sweldens, ''The lifting scheme: a new philosophy in biorthogonal wavelet constructions,'' in *Wavelet Applications in Signal and Image*

Processing III, A. F. Laine and M. Unser, Eds., *Proc. SPIE* **2569**, 68–79 (1995).
8. I. Daubechies and W. Sweldens, ''Factoring wavelet transforms into lifting steps,'' Technical Report, Bell Laboratories, Lucent Technologies (1996).
9. G. Deslauriers and S. Dubuc, ''Interpolation dyadique,'' in *Fractals, dimensions non entières et applications*, pp. 44–55, Masson, Paris (1987).
10. W. Sweldens, ''The lifting scheme: a custom-design construction of biorthogonal wavelets,'' *Appl. Comput. Harmon. Anal.* **3**(2), 186–200 (1996).
11. G. Fernández, S. Periaswamy, and W. Sweldens, ''LIFTPACK: A software package for wavelet transforms using lifting,'' in *Wavelet Applications in Signal and Image Processing IV*, M. Unser, A. Aldroubi, and A. F. Laine, Eds., *Proc. SPIE* **2825**, 396–408 (1996).
12. W. Sweldens, ''The lifting scheme: a construction of second generation wavelets,'' *SIAM J. Math. Anal*. (in press).
13. P. Schröder and W. Sweldens, ''Spherical wavelets: efficiently representing functions on the sphere,'' in Computer Graphics Proc. (SIGGRAPH 95), pp. 161–172, ACM Press (1995).
14. W. Sweldens and P. Schröder, ''Building your own wavelets at home,'' in *Wavelets in Computer Graphics, ACM SIGGRAPH Course Notes*, pp. 15–87, ACM Press (1996).
15. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, ''Wavelets transforms that map integers to integers,'' Technical Report, Department of Mathematics, Princeton University (1996).
16. H. Chao and P. Fisher, ''An approach to fast integer reversible wavelet transforms for image compression,'' Technical Report, Computer and Information Science, Inc. (1996).
17. S. G. Mallat, ''A theory for multiresolution signal decomposition: the wavelet representation,'' *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(7), 674–693 (1989).
18. S. G. Mallat, ''Zero crossings of a wavelet transform,'' *IEEE Trans. Inf. Theory* **37**(4), 1019–1033 (1991).
19. S. Mallat and S. Zhong, ''Characterization of signals from multiscale edges,'' *IEEE Trans. Pattern Anal. Mach. Intell*. **14**, 710–732 (1992).
20. S. Mallat and S. Zhong, ''Wavelet transform maxima and multiscale edges,'' in *Wavelets and Their Applications*, M. B. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael, Eds., pp. 67–104, Jones and Bartlett, Boston (1992).
21. W. Förstner, ''A framework for low level feature extraction,'' in *Computer Vision—ECCV'94*, Vol. 2, J. O. Eklundh, Ed., pp. 283–294, Springer-Verlag, Berlin (1994).
22. A. Bushc, ''A common framework for the extraction of lines and edges,'' *Int. Arch. Photogram. Remote Sens*. **31**, 88–93 (1994).
23. H. Voorhees and T. Poggio, ''Detecting blobs and textons in natural images,'' in *Proc. DARPA Image Understanding Workshop*, pp. 892–899, Los Angeles, CA (1994).
24. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, 2nd ed., Cambridge University Press (1993).
25. O. Henricsson and F. Heitger, ''The role of key-points in finding contours,'' in *Computer Vision—ECCV'94*, Vol. 2, J. O. Eklundh, Ed., pp. 371–383, Springer-Verlag, Berlin (1994).
26. G. Fernández, ''Multiresolution feature extraction using lifting,'' Master's Thesis, University of South Carolina, Columbia (1997).
27. W. E. L. Grimson, ''Object recognition by computer: the role of geometric constraints,'' The MIT Press (1990).
28. M. B. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael, Eds., *Wavelets and Their Applications*, Jones and Bartlett, Boston (1992).



Gabriel Fernández received his BS in systems engineering in Caracas, Venezuela, in 1992. After working as a field engineer for Silicon Graphics in Venezuela for a year and a half, he began graduate studies in the United States. In May 1997, he received his MS in computer science from the University of South Carolina. His research interests include applications of wavelets to signal and image processing looking for better and faster implementations of the discrete wavelet transform using the lifting scheme. He is currently a design engineer for magnetic resonance software applications with GE Medical Systems.



Terrance Huntsberger received his PhD in physics from the University of South Carolina in 1978, where he currently directs the Intelligent Systems Laboratory and is an associate professor in the Department of Computer Science. His current research interests include wavelet-based image analysis, computer graphics, and parallel simulation of physical processes. He is a member of the SPIE, ACM, INNS, and the IEEE Computer Society.